Computational Plant (Cplant™)

Ron Brightwell

Sandia National Labs

Scalable Systems Integration Department

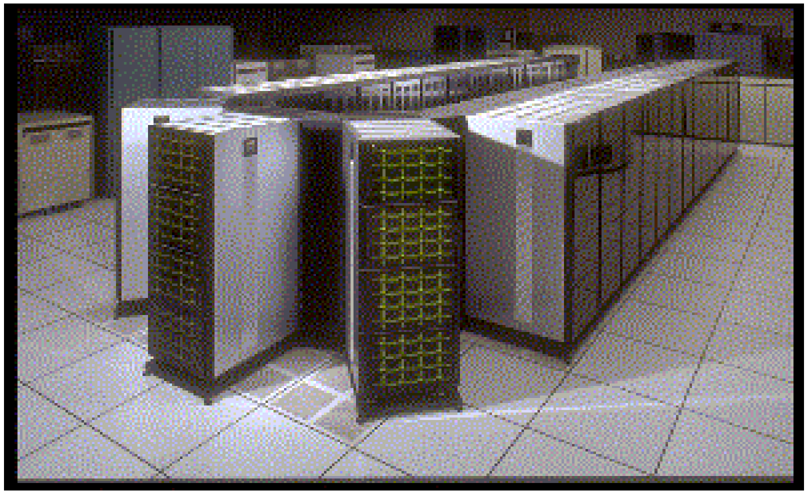
bright@cs.sandia.gov

Outline

- Cplant™ Hardware
- Cplant™ Runtime System
- Application Peformance

System Software R&D at Sandia

- Intel Paragon
  - 1890 compute nodes
  - 3680 i860 cpu's
  - 143/184 GFLOPS
  - 175 MB/sec network
- SUNMOS lightweight kernel
  - High performance compute node OS for distributed memory MPP's
  - Deliver as much performance as possible to apps
  - Small footprint
  - Started in January 1991 on the nCUBE-2 to explore new message passing schemes and high-performance I/O
  - Ported to Intel Paragon in Spring of 1993



System Software R&D (cont'd)

- Intel ASCI Red
  - 4576 compute nodes
  - 9472 Pentium II CPU's
  - 2.38/3.21 TFLOPS
  - 400 MB/sec network
- Cougar lightweight kernel
  - Multiprocess support
  - Modularized (QK, PCT)
  - Developed on nCUBE-2 in 1993
  - Ported to Intel Paragon in 1995
  - Ported to Intel TFLOPS in 1996 (Cougar)
  - Portals 1.0
    - User/Kernel managed buffers
  - Portals 2.0
    - Avoid buffering and memory copies

Why Cplant™?

- Modeling and simulation, essential to stockpile stewardship, require significant computing power
- Commercial supercomputers seemed to be a dying breed
- Pooling of large SMP's is expensive and more complex
- Commodity PC market is closing the performance gap
- Web services and e-commerce are driving high-performance interconnect technology

What is Cplant™?

- Cplant™ is a concept
  - Provide computational capacity at low cost
  - Build MPPs from commodity components
  - Follow ASCI Red model and architecture
- Cplant™ is an overall effort:
  - Multiple computing systems in NM & CA
  - Multiple projects
    - Portals 3.x message passing (with UNM and others)
    - Cluster Infrastructure Toolkit (with HPTi)
    - System integration & test
    - Operations & management
- Cplant™ is a software package
  - Available under the GNU LGPL

Cplant™ Approach

- Hybrid approach combining commodity cluster technology with MPP technology
- Emulate the Intel ASCI Red environment
  - Partition model (functional decomposition)
  - Space sharing (reduce turnaround time)
  - Scalable services (allocator, loader, launcher)
  - Complete compute node resource dedication
- Use Existing Software when possible
  - Red Hat distribution, Linux/Alpha
  - Software developed for ASCI Red

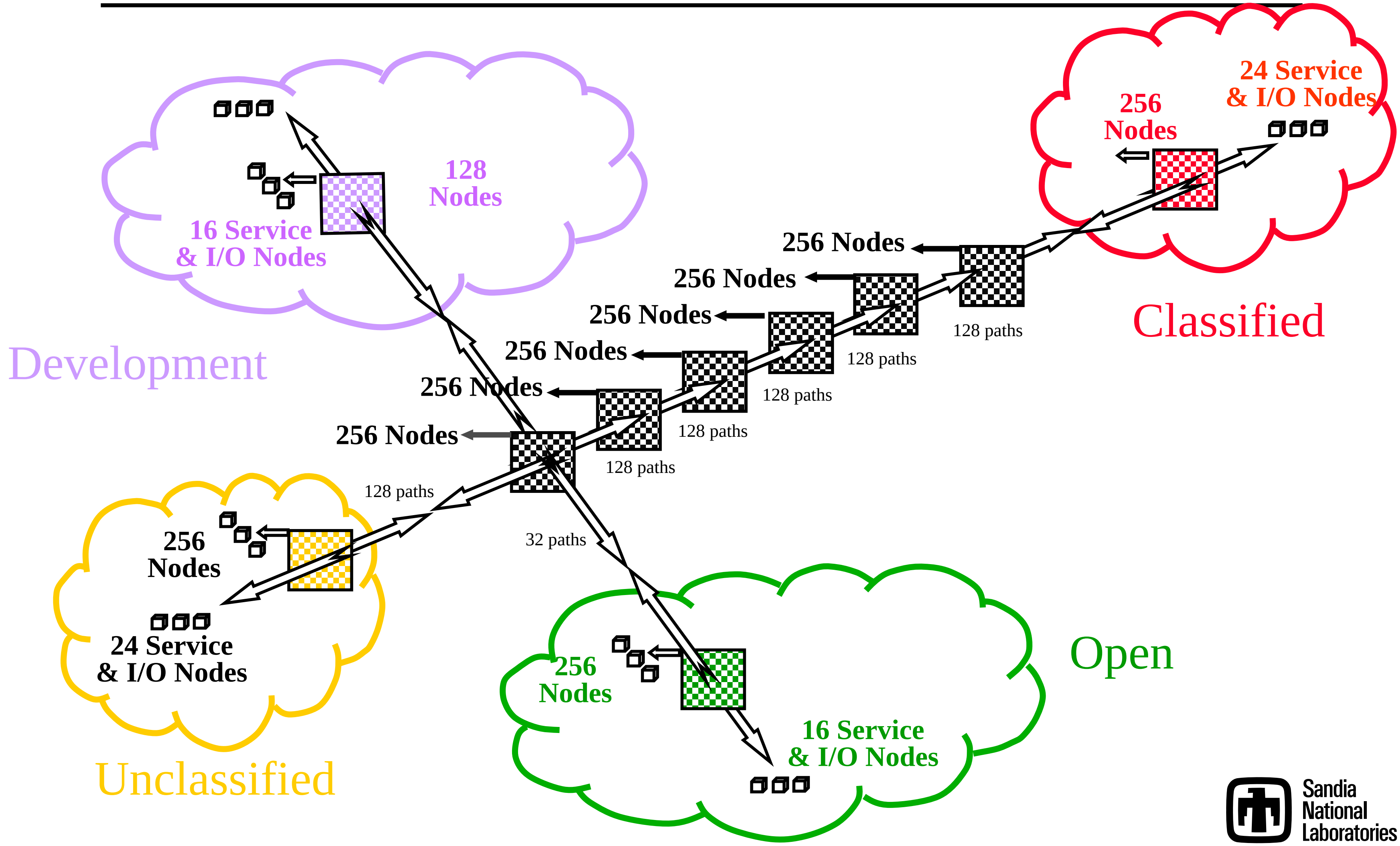
Cplant™ Systems (SNL/NM)

	Vendor	Compaq	Compaq	Compaq	Compaq	Compaq	Digital	Dell		
	Model	DS10L	DS10L	DS10L	XP1000	XP1000	500au	PowerEdge		
	CPU	Alpha EV67	Alpha EV6	Alpha EV6	Alpha EV6	Alpha EV6	Alpha EV56	Pentium III		
	CPU Freq.	617 MHz	466 MHz	466 MHz	500 MHz	500 MHz	500 MHz	1 GHz		
	Memory	1 GB	256 MB	1 GB	256 MB	1 GB	192 MB	1 GB		
Name	**Network**	**Number of Nodes**							**Interconnect**	**Deployment**
Ross	SRN	256							Myrinet LANai-7,9	Production
Ronne	SCN				256				Myrinet LANai-7,9	Production
West	SON			96		160			Myrinet LANai-7,9	Production
Center	Switchable		1536						Myrinet LANai-7,9	Production
Alaska	SRN						258		Myrinet LANai-4	Production
Zermatt	SRN				128				Myrinet LANai-7,9	Development
Iceberg2	SON				14				Myrinet LANai-7	Development
Iceberg	SRN						28		Myrinet LANai-4	Development
Quadrics	SRN							4	Quadrics ELAN-3	Development
Total Peak (GFLOPS)		315.90	1431.55	89.47	398.00	160.00	286.00	4.00		
								2684.93		

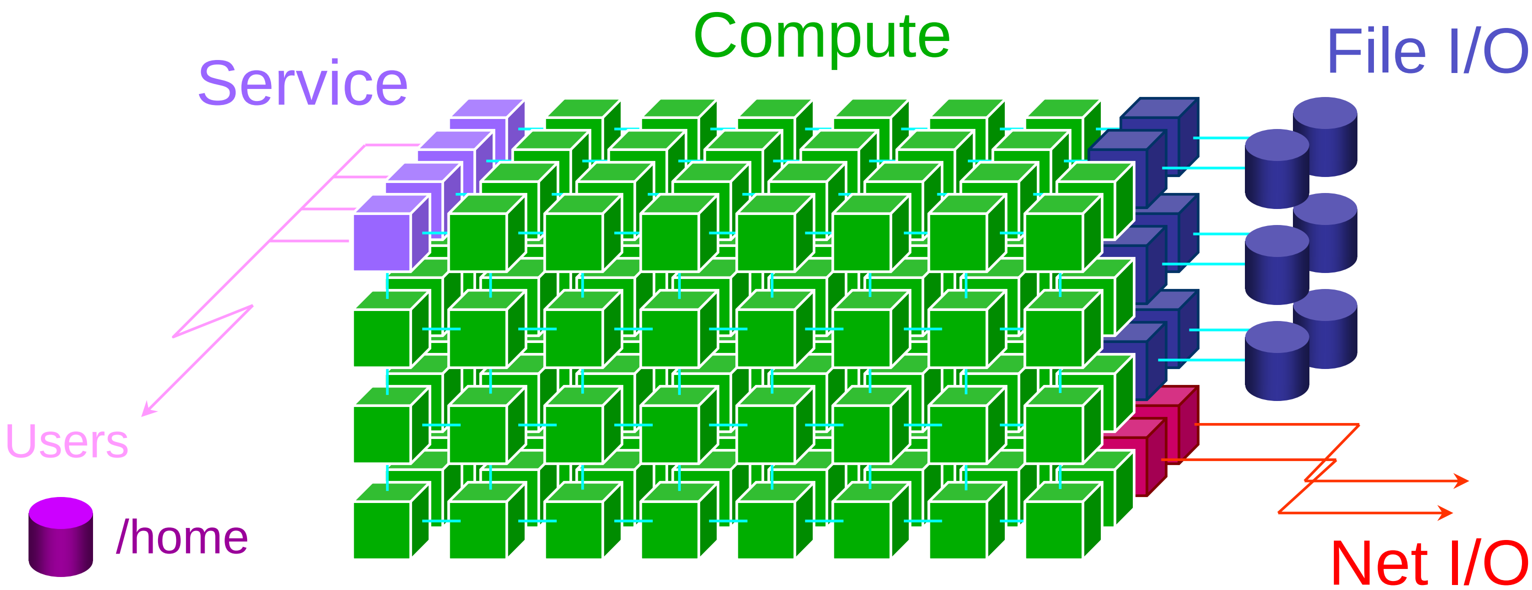
Antarctica

- 1792+ Compaq DS10L Slates
  - 466MHz EV6, 256 MB RAM
- 590 Compaq XP1000s
  - 500 MHz EV6, 256 MB RAM
- Myrinet 33MHz 64bit LANai 7.x and 9.x
- Myrinet Mesh64 switches
- Classified, unclassified, open, and development network heads

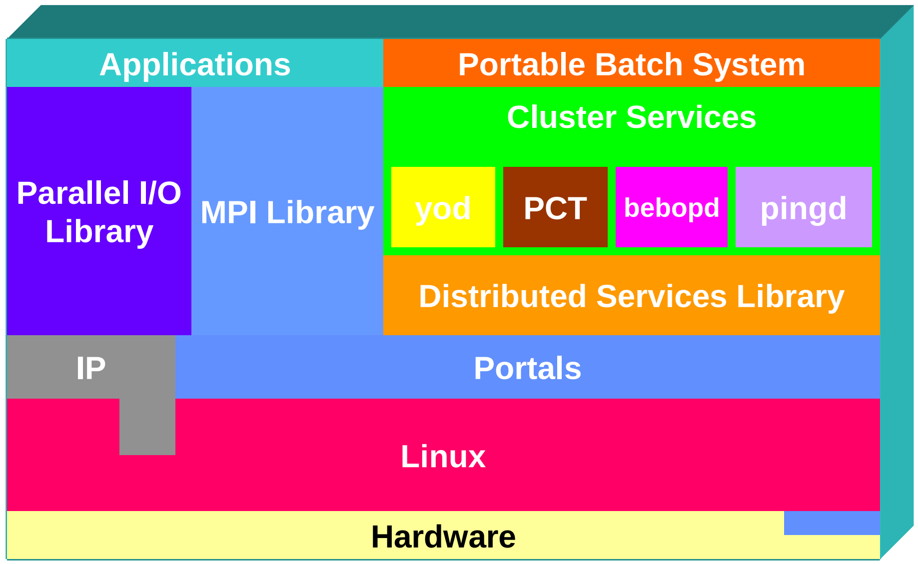
Antarctica’s Center Can Connect to Four Different Heads

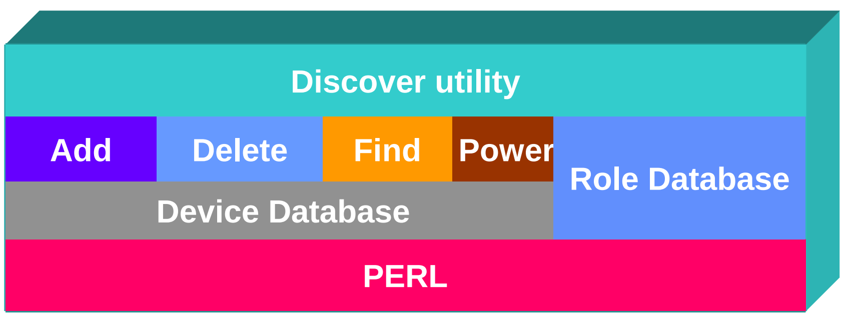


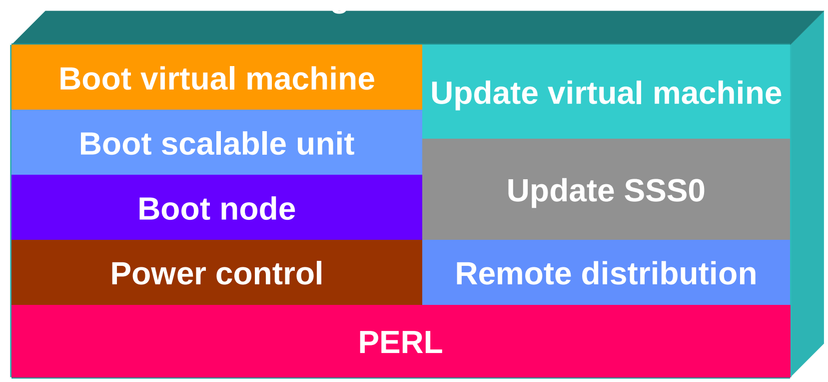
Conceptual Partition Model



Cplant™ Software







Runtime System Components

- Yod (xnc++)
  - Service node parallel job launcher
- Yod2
  - Job launcher for dynamic process creation
  - Not yet deployed in production
- Bebopd (Better Engineered Bag Of PCs Daemon)
  - Compute node allocator
- PCT (Process Control Thread)
  - Compute node daemon
- pingd/showmesh
  - Compute node status tools
- PBS
  - Batch scheduler

Runtime System (cont'd)

- Yod
  - Contacts compute node allocator
  - Launches the application into the compute partition
  - Redirects all application I/O (stdio, file I/O)
  - Makes any filesystem visible in the service partition visible to the application
  - Redirects any UNIX signals to compute node processes
  - Allows user to choose specific compute nodes
  - Can launch multiple different binaries
  - Displays launch timing information
  - Same basic interface as SUNMOS and Cougar

Runtime System (cont'd)

- PCT
  - Contacts bebopd to join compute partition
  - Forms a spanning tree with other PCT's to fan out the executable, shell environment, signals, etc.
  - Puts executable in a RAM disk
  - *fork()*'s, *exec()*'s, and monitors status of child process
  - Cleans up after parallel job

Runtime System (cont'd)

- Bebopd
  - Accepts requests from PCT's to join the compute partition
  - Accepts requests from yod for compute nodes
  - Accepts requests from pingd for status of compute nodes
  - Coordinates scheduling with PBS server
  - Allows for multiple compute partitions

Runtime System (cont'd)

- Pingd
  - Displays list of available compute nodes
  - Displays list of compute nodes in use
  - Displays owner, elapsed time of jobs
  - Allows users to kill their jobs
  - Allows administrators to kill jobs and free up specific nodes
  - Allows administrators to remove nodes from the compute partition
- Showmesh
  - Massages pingd output into TFLOPS-like showmesh

Runtime System (concl'd)

- PBS
  - Enhanced version of OpenPBS
  - Added non-blocking I/O for fault tolerance
  - PBS Moms and Server only run in the service partition
  - Added new attribute – "nodes"
  - Contacts bebopd to get a list of nodes to give to yod

User-Level Software

- Redirected standard C and I/O libraries
  - Catch some system calls and let yod handle them
  - Uses a RPC library over Portals 3.0
- Distributed services library
  - Used by for communication between runtime system components (yod, pct, bebopd)
  - Implemented over Portals 3.0
- Puma library
  - Implements *dclock()* and others for compatibility with Puma
- Startup code
  - Initializes the parallel environment for a process

User-Level Software (cont'd)

- MPI library
  - Portals 3.x device layer for MPICH 1.2.0
  - Implements peer communication only
- Dynamic allocation library
  - New code to support MPI-2 dynamic process creation functionality
  - Not yet deployed in production
- Job library
  - Allows for user-implemented job launcher
- Portals 3.x library
  - Basic peer communication functions

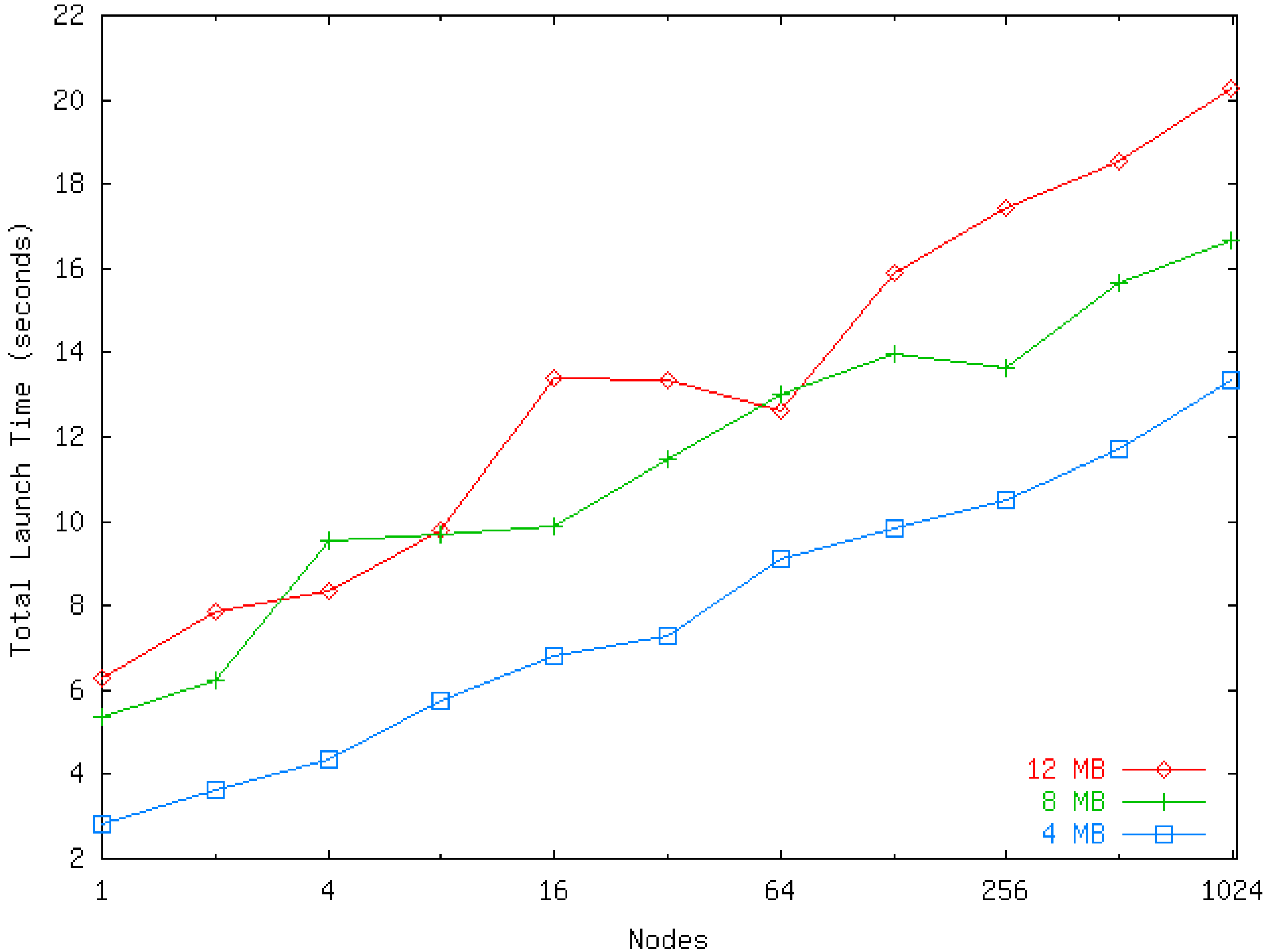
Kernel-Level Software

- Minor patches to Linux for memory locking and memory mapping
- Address cache module (unused)
  - Caches virtual-to-physical mappings for Portals 3.x
- cTask module
  - Runtime system mappings for processes
  - Process cleanup
- Portals 3.x module
  - Implements Portals 3.x functionality
- RTS/CTS module
  - Myrinet device driver
  - Reliability and flow control
- MyrIP module
  - Provides IP packets over Myrinet

Device-Level Software

- Myrinet Control Program
  - Firmware running on LANai processor on NIC
  - Packet engine

Cplant™ Can Launch 1010-Node Jobs in Seconds



Design Issues

- Two ways to move executable to compute nodes
  - Pull executable to compute nodes
    - Requires some intelligence in the filesystem
    - Filesystems can't handle N-to-1 reads
  - Push executable to compute nodes
    - No filesystem dependency
    - Easier to implement
- Need to start processes in parallel
- Support for other programming models
  - Job launch should not be specific to the programming model
- Fault detection

Design Issues (cont'd)

- Bebopd is a single point of failure
  - No new jobs runs if bebopd goes away
  - Distributed bebopd
    - Failure only affects part of the cluster
    - Haven't needed to do it yet
  - Bebopd checkpoints the state of the machine and can be restarted

Emphasis on Reliability

- More nodes, more users, more applications lead to more stress on the system
- Myrinet issues
  - GM mapper limitations
    - Each new cluster exceeded the number of nodes the mapper could handle
    - Entire cluster must be up and running
  - Non-deadlock-free routes
    - Code for routing algorithm gave only shortest path routes
  - Reliability
    - Bit error rate orders of magnitude higher than advertised
    - Storms of multi-bit errors
    - Mis-routed packets, corrupted headers, corrupted data

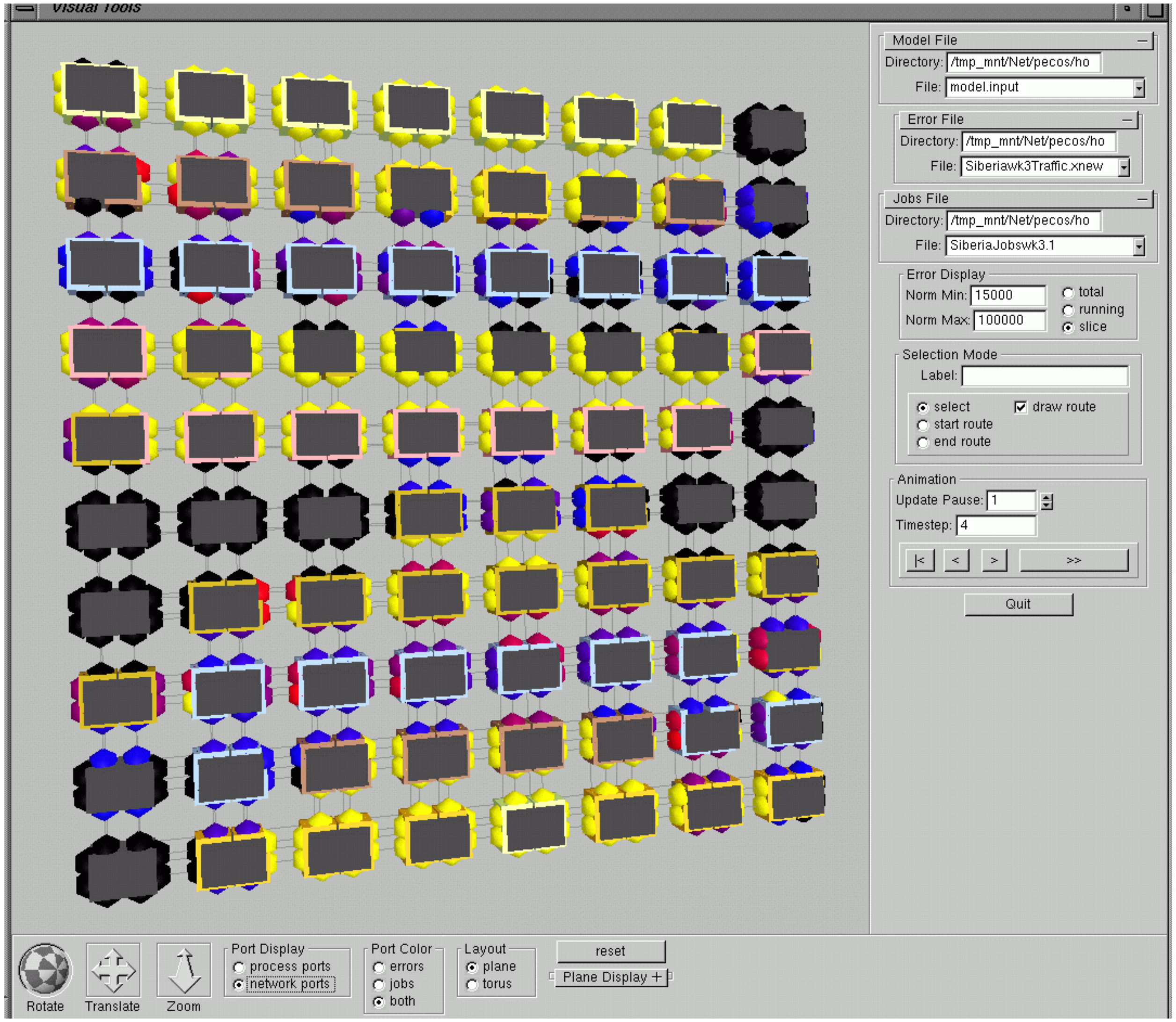
Emphasis on Reliability (cont'd)

- Runtime system issues
  - Most problems related to message passing
    - Runtime utilities must recover from network errors
  - Problems show up as
    - Failure to start parallel job
    - Utilities become uncommunicative
    - Compute nodes become unreachable
    - Allocator becomes unresponsive

Addressing Message Passing Reliability and Robustness

- Added error detection/correction to Myrinet driver
- Implemented Myrinet switch monitoring software
- Implemented switch error visualization tool
- Fixes to the network reliability protocol
  - Fixes to message sequencing bug
  - Propagation of failures up the network stack
- Portals
  - Fixes to event ordering semantics
  - Defined transport failure semantics
  - Enhancement for more scalable buffering of MPI unexpected messages

Switch Error Visualization Tool

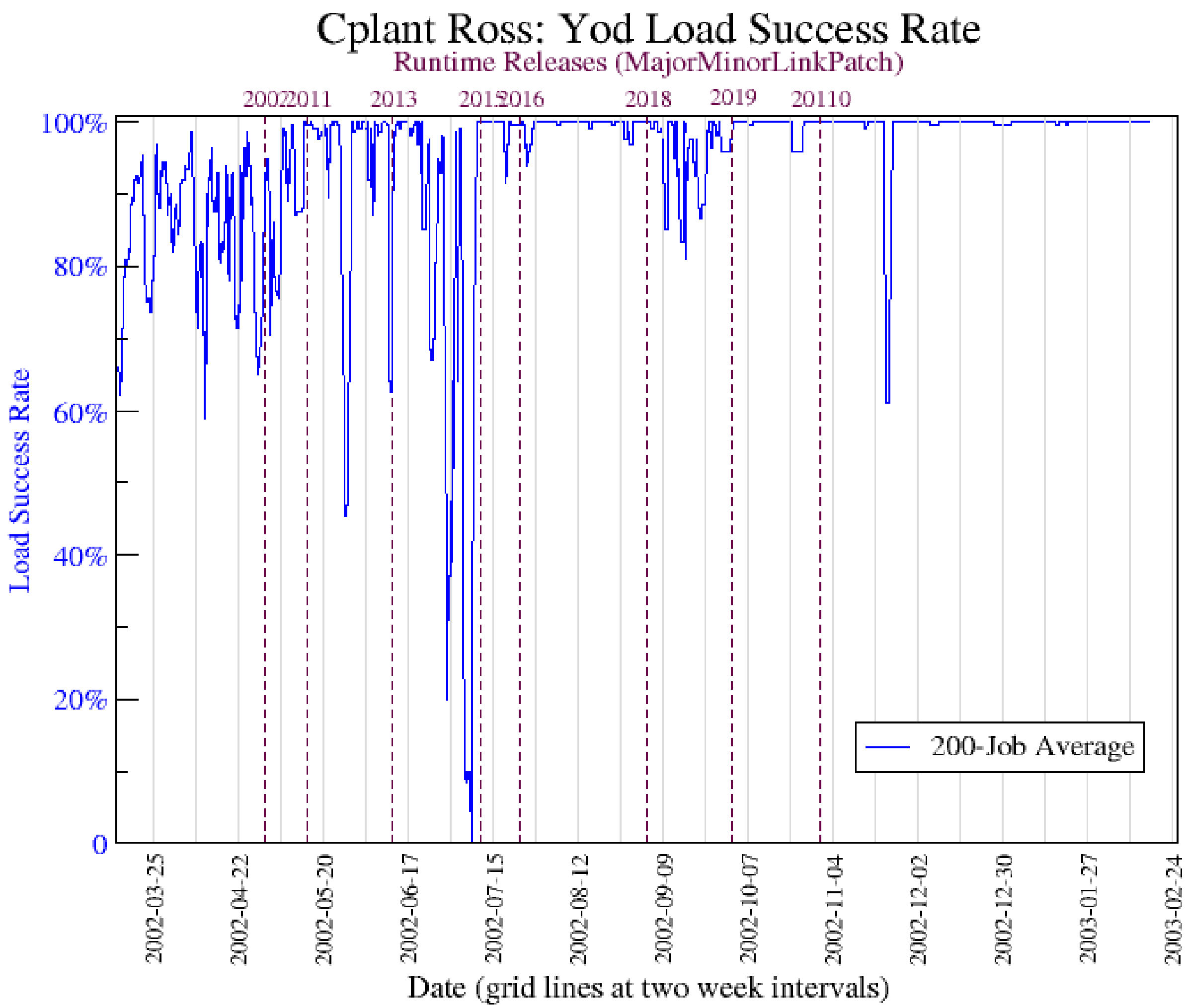


Sandia National Laboratories

Addressing Runtime System Reliability and Robustness

- Stripped-down load protocol
  - Enhancement to avoid non-scalable operations
  - Nodes automatically pruned during load failures
- Enhancements to compute node allocator
  - Single point of failure
  - Throttling of messages from compute nodes
  - Allocator now stateful
- Changes to allow centralized runtime logging
- Issue tracking system

Cplant™ Robustness



Salinas on Cplant™

ASCI/Red Hardware

- 4640 compute nodes
  – Dual 333 MHz Pentium II Xeons
  – 256 MB RAM
- 400 MB/sec bi-directional network links
- 38x32x2 mesh topology
- Red/Black switchable
- First machine to demonstrate 1+ TFLOPS
- 2.38/3.21 TFLOPS
- Deployed in 1997

ASCI/Red Compute Node Software

- Puma lightweight kernel
  - Follow-on to Sandia/UNM Operating System (SUNMOS)
  - Developed for 1024-node nCUBE-2 in 1993 by Sandia/UNM
  - Ported to 1800-node Intel Paragon in 1995 by Sandia/UNM
  - Ported to Intel ASCI/Red in 1996 by Intel/Sandia
  - Productized as “Cougar” by Intel

ASCI/Red Software (cont'd)

- Puma/Cougar
  - Space-shared model
  - Exposes all resources to applications
  - Consumes less than 1% of compute node memory
  - Four different execution modes for managing dual processors
  - Portals 2.0
    - High-performance message passing
    - Avoid buffering and memory copies
    - Supports multiple user-level libraries (MPI, Intel N/X, Vertex, etc.)

Salinas

- General-purpose, finite element structural dynamics code for massively parallel computers
- Currently offers
  - Static analysis
  - Direct implicit transient analysis
  - Eigenvalue analysis for computing modal response, modal superposition-based frequency response, and transient response

Salinas (cont'd)

- Includes extensive library of
  - Standard one-, two-, and three-dimensional elements
  - Nodal and element loading
  - Multi-point constraints

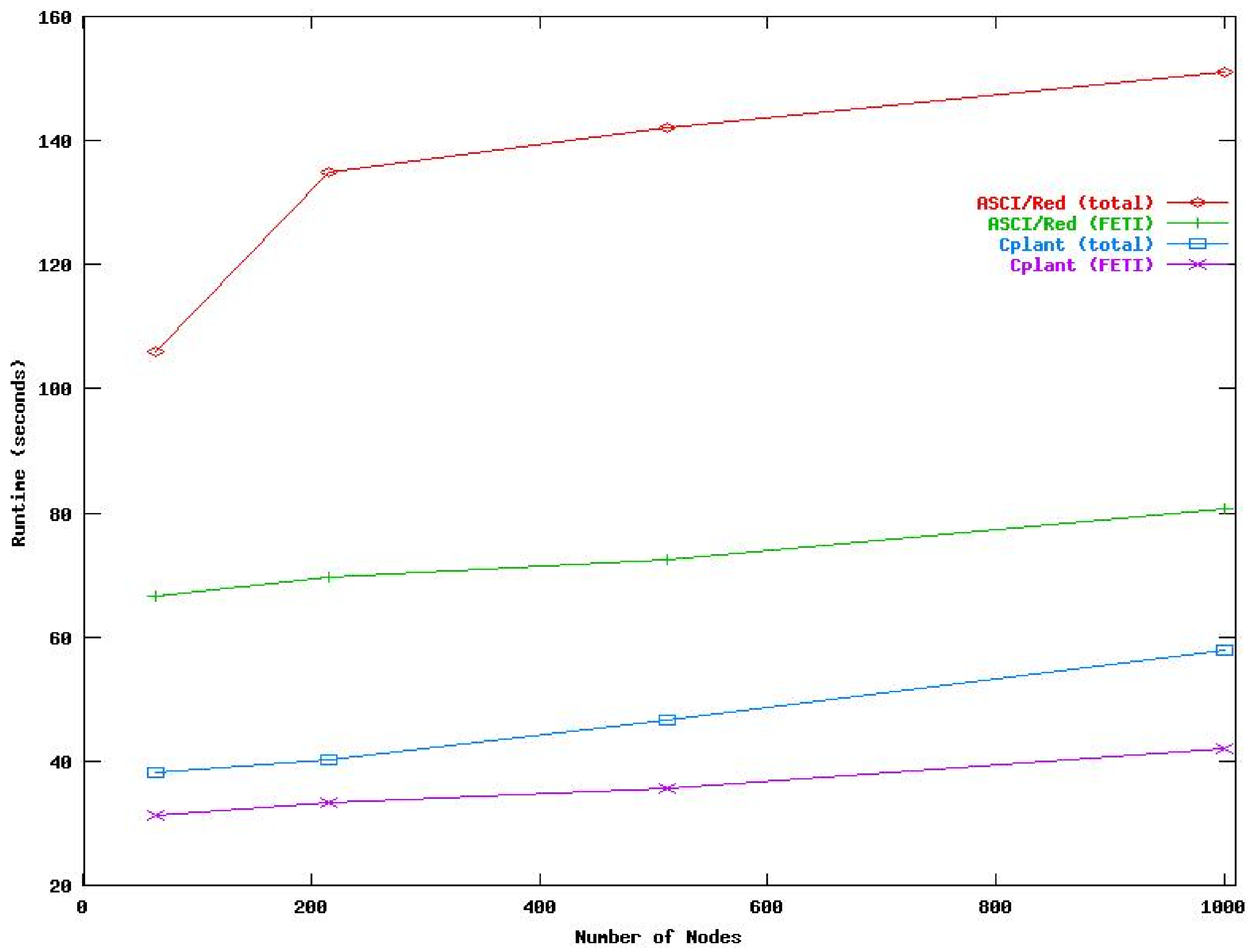
Salinas (cont'd)

- Solves systems of equations using an iterative multilevel solver specifically designed to exploit massively parallel machines
  - Finite Element Tearing and Interconnect (FETI)
  - Mature
    - Versions used in commercial finite element packages
  - Scalable
    - As the number of unknowns increases and the number of unknowns per processor stays constant, time to solution stays constant
  - Accurate
    - Convergence rate does not deteriorate as the iterates converge

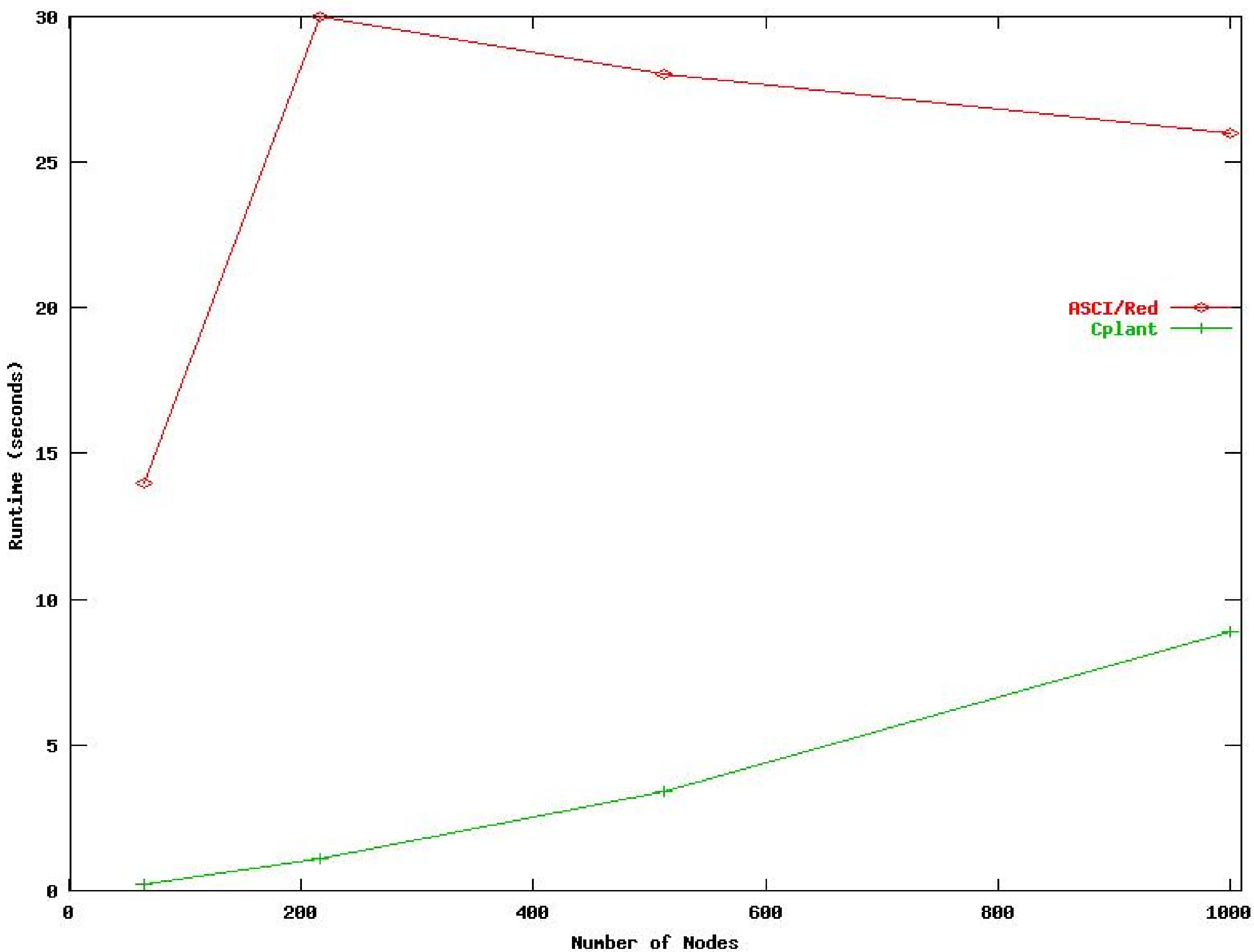
Salinas Sample Problem

- Small problem size
  - Only bout 3 MB per node
- Stresses the system more than larger problems
  - Ratio of computation to communication is larger
  - Higher frequency of message passing
- Good indicator of scaling efficiency for larger problems
- Dedicated time on Cplant™
- Non-dedicated time on ASCI/Red using a single processor per node
- Average of five runs

Salinas is 2.5x Faster on Cplant™ at 1000 nodes



I/O Time Is Not Scaling As Well on Cplant™

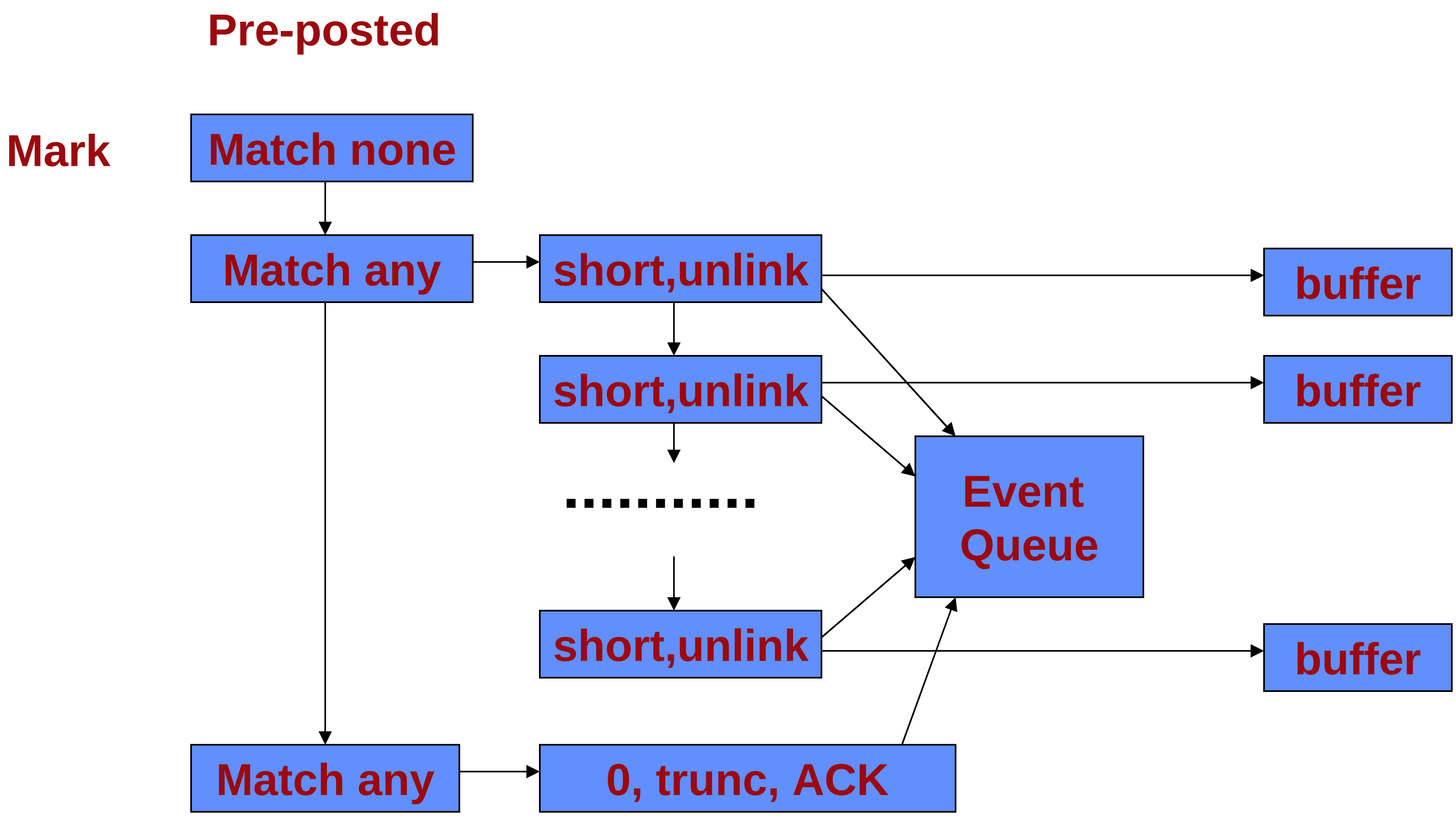


Scaling Issue on Cplant™

- MPI resource exhaustion at several hundred nodes
- "Too many MPI unexpected messages"
  - AKA "Not enough posted receives"
- Short message protocol for MPI is eager
- Unexpected messages are buffered at the receiver
- Initial MPI implementation set aside 1024 8 KB buffers
- A single message of any size consumes a buffer

- MPI_Gather() in MPICH 1.2.0 is implemented via N-to-1 algorithm
- Quick workaround was to add an MPI_Barrier() to make MPI_Gather() synchronous

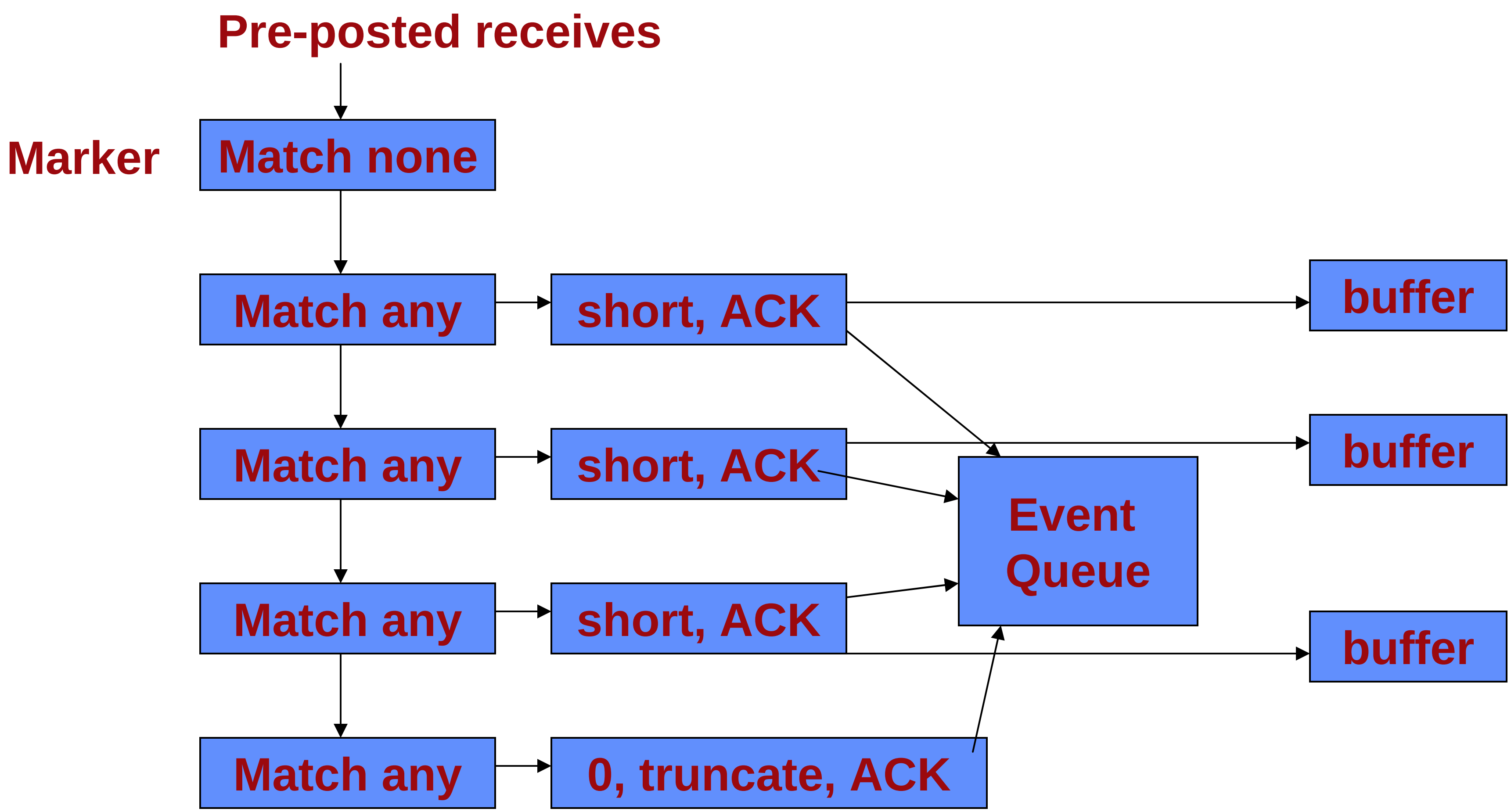
Previous Strategy for Unexpected Messages



Limitations

- Limited number of unexpected messages allowed due to kernel (or NIC) memory resources
- Any size unexpected message consumes an unexpected message slot, even zero-length
- Unexpected message limit based on count rather than size
- Consumes a significant amount of Portals resources
  - 1025 memory descriptors

Current Strategy



Advantages

- More efficient use of unexpected message memory
  - A zero-length message doesn’t consume any memory
  - Limitation becomes space rather than count
- Uses only a few Portals resources
  - Four memory descriptors versus 1025
- More efficient for NIC-based implementations

As for Salinas…

- Change to MPI library had minimal effect on performance
- Overhead of extra MPI_Barrier() operation to synchronize MPI_Gather() operation is negligible

Salinas Summary

- A commodity Linux cluster is able to sustain competitive performance for a real-world code out to 1000 nodes
- Cplant™ is a viable, reliable, large-scale platform
- Issues with network resources become important as applications scale

Ongoing Runtime System Work

- Intelligent allocator
  - Try to account for network topology or routes
  - Ideal allocator would allocate contiguous nodes
  - Measure impact on load time
- Dynamic process creation
  - Support for MPI-2 dynamic process creation functions
- Multiprocessor support
  - Current environment supports one process per node
- Multithreaded support
  - Support using pthreads in an application process
- Library API for runtime system interaction
  - Host library for custom allocator

Licensing

- Cplant™ source code released under the GNU LGPL
  - 1400+ downloads since April 19, 2001
- Cplant™ source code licensed to Unlimited Scale, Inc.
  - Intended to be base technology for initial product
  - Sandia has a small equity in USI

Acknowledgments

- **Salinas**
  - Manoj Bhardwaj, Garth Reese (SNL)
- **Portals**
  - Barney Maccabe (University of New Mexico)
  - Peter Braam (Cluster File Systems, Inc.)

http://www.cs.sandia.gov/cplant

http://sf.net/project/sandiaportals